Expert Systems with Applications 40 (2013) 5456–5465

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Rule-based expert systems to support step-by-step guidance in algebraic problem solving: The case of the tutor PAT2Math

Patricia A. Jaques [a,*], Henrique Seffrin [a], Geiseane Rubi [a], Felipe de Morais [a], Cássio Ghilardi [a], Ig Ibert Bittencourt [b], Seiji Isotani [c]

[a] *PIPCA – UNISINOS, Av. Unisinos, 950 Bairro Cristo Rei, CEP 93.022-000 São Leopoldo, Brazil*
[b] *NEES – IC – UFAL, Campus A.C. Simões, BR 104, Norte, km 97, Cidade Universitária, CEP 57072-970 Maceio, AL, Brazil*
[c] *ICMC – University of São Paulo, Avenida Trabalhador São-carlense, 400 Centro, CEP 13566-590 Sao Carlos, SP, Brazil*

ARTICLE INFO

Keywords:
Intelligent tutoring systems
Expert systems
Math learning
Algebra
Equations

ABSTRACT

In order for an Intelligent Tutoring System (ITS) to correct students' exercises, it must know how to solve the same type of problems that students do and the related knowledge components. It can, thereby, compare the desirable solution with the student's answer. This task can be accomplished by an expert system. However, it has some drawbacks, such as an exponential complexity time, which impairs the desirable real-time response. In this paper we describe the expert system (ES) module of an Algebra ITS, called PAT2Math. The ES is responsible for correcting student steps and modeling student knowledge components during equations problem solving. Another important function of this module is to demonstrate to students how to solve a problem. In this paper, we focus mainly on the implementation of this module as a rule-based expert system. We also describe how we reduced the complexity of this module from $O(n^d)$ to $O(d)$, where n is the number of rules in the knowledge base, by implementing some meta-rules that aim at inferring the operations students applied in order to produce a step. We evaluated our approach through a user study with forty-three seventh grade students. The students who interacted with our tool showed statistically higher scores on equation solving tests, after solving algebra exercises with PAT2Math during an approximately two-hour session, than students who solved the same exercises using only paper and pencil.

1. Introduction

Intelligent Tutoring Systems (ITSs) have shown promising results when applied as a supplemental classroom learning tool (Koedinger, Anderson, Hadley, & Mark, 1997; Nicaud, Bittar, Chaachoua, Inamdar, & Maffei, 2006; Nicaud, Bouhineau, & Huguet, 2002). Large-scale experiments in high-schools demonstrated that ITSs can improve students learning (Koedinger et al., 1997; Koedinger & Sueker, 1996). The success of this type of educational software is due to the fact that it can offer important features to personalize the learning processes such as one-on-one learning, immediate personal feedback, demonstration of problem solving when students are having difficulty, and assessment of students' skills.

Vanlehn (2006) describes the tutor as having two loops. The *outer loop* is responsible for deciding the sequence of exercises or problems for students to work on. The *inner loop* provides step-by-step guidance during problem solving activity.

In order to provide immediate feedback in the inner loop, an ITS' architecture is generally composed of an expert system module (ES) that is able to solve the same type of exercises that students should do in multiple ways. Thus, for each step of the problem the system compares the answer provided by a student with the expert system's solutions (an answer may have several correct solutions) and checks whether they are equivalent. If the ES can generate or test all possible solutions to a given problem, then it can identify when a student has taken an incorrect path to solve the given problem and offer immediate feedback (generally colored labels are presented to indicate whether or not the student's answer for a given step is correct) (Heffernan, Koedinger, & Razzaq, 2008). Some tutors provide additional resources (e.g. explanations or hints) when students are having difficulty or solving the problem or arriving at the correct answer.

In fields such as math and physics, the knowledge is usually implemented as a rule in the form: "*if* ⟨*condition is true*⟩ *then* ⟨*do action A*⟩". Each rule represents an operation that can be applied in a step to solve a problem. The ES inference engine scans the base searching for rules to be triggered, i.e. rules whose conditions are satisfied by the current step of the solution.

* Corresponding author. Tel.: +55 51 3591 1100x1626.

E-mail addresses: pjaques@unisinos.br, patricia.jaques@gmail.com (P.A. Jaques), hseffrin@hotmail.com (H. Seffrin), geiserubi@gmail.com (G. Rubi), felipedemoraisfm@hotmail.com (F. de Morais), cassioghilardi@hotmail.com (C. Ghilardi), ig.ibert@ic.ufal.br (I.I. Bittencourt), sisotani@icmc.usp.br (S. Isotani).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.04.004

In addition to providing appropriate feedback to students, the rules also contain information about the related knowledge components. For example, a rule can represent the process (knowledge component or skill) of "subtracting an integer b on both sides of the equation". Thus, when the ES module corrects a student step solution, it is able to provide the student model with information about the skills necessary to solve that step. The student model uses this information to infer which skills students have mastered and which they need to practice more. This allows the tutor to create personalized hints in the inner loop and also to select more appropriate exercises for students to solve (outer loop).

Although there are well known algebra tutors (Chaachoua, Nicaud, Bronner, & Bouhineau, 2004; Cohen, Beal, & Adams, 2008; Koedinger & Sueker, 1996; Melis, Goguadze, Libbrecht, & Ullrich, 2009), few of them have an expert system module that is able to solve exercises and provide step-by-step guidance (Chaachoua et al., 2004; Koedinger & Sueker, 1996), an essential feature for a learning system to be classified as an ITS, according to (Vanlehn, 2006). Furthermore, previous work does not explore in detail how to implement an expert system module, which artificial intelligence knowledge representation format needs to be used, when an inference mechanism should be trigged and how to solve some inherent computational complexity problems.

This paper presents the ES of the algebra tutor PAT2Math. PAT2-Math is an intelligent tutor system that teaches students how to solve linear and quadratic equations. It is a web system implemented in Java, which allows students to use it in any computer or platform with Internet access. PAT2Math is composed of an algebra editor (PATequation), which assists students in solving equations.

The ES has an essential role in PATequation; it is responsible for providing immediate feedback to students at every step of their problem solving. Our main goal is to present this module knowledge and explain how to the ES implements problem solving and provides the student with step-by-step guidance. We describe how to reduce the complexity of this module from $O(n^d)$ to $O(d)$, where n represents the number of rules in the knowledge base, by using meta-rules that guide the inference of the operations student applied to produce a step. We finish this paper by presenting the results of a user study we conducted with forty-three 7th grade students who interacted with PATequation for three classes.

This paper is organized as follows. Section 2 describes problem solving under a pedagogical perspective. Section 3 presents the current state of the art in Algebra Intelligent Tutoring Systems. In Section 4, we explain the main artificial intelligence techniques used to develop ITS expert systems. In Section 5, we describe PAT2-Math, the Algebra Tutor that our research group is developing. PATequation, the problem solving editor of PAT2Math, is presented in Section 6. The ES responsible for providing step-by-step guidance in PATequation is described in Section 7. The experiment design and results are reported in Section 8. Finally, Section 9 presents our conclusions.

2. Solving algebraic problems

An Algebra task is generally a word problem for the student to solve. An algebraic word problem consists of one or more sentences representing a situation or a story, where the student needs to understand the elements in order to generate a mathematical model to represent it. The model consists of one or more equations that the pupil should solve in order to obtain the numerical values that are the solution of the problem (Gama, 2004). Take for example the word problem below (adapted from Munem & West (2003, p. 107)):

"A computer store sells desktop and laptop computers. Due to space considerations, the number of laptops in inventory is seven less than twice the number of desktops in stock. How many desktops does the store have if it has a total of 272 computers?"

The process of solving a word problem has two phases (Mayer, 1999; Polya, 2004): (i) the Problem Representation (also called Symbolization (Heffernan, Koedinger, & Razzaq, 2008)), and (ii) the Problem Solution. While the former concerns to the transformation of algebra word problems into a system of equations, the second encompasses the process of solving these equations using algebraic operations.

For example, for the word problem that we previously presented, the student could provide the following solution:

$$x + (2x - 7) = 272 \tag{1}$$
$$3x - 7 = 272 \tag{2}$$
$$3x = 279 \tag{3}$$
$$x = 93 \tag{4}$$

In the example above, line (1) refers to the process of Problem Representation, and lines (2–4) to the Problem Solution phase. As shown in the above solution, solving a task involves several steps. Each line provided by the student in the above solution is a step.

A step can involve the correct use of one or more Knowledge Components (KC) (also called knowledge units (Aleven, McLaren, Sewall, & Koedinger, 2009)). It comprises any unit into which the knowledge can be broken down, such as rules, concepts, facts, and procedures (Vanlehn, 2006). For instance, in order to arrive at line (2) in the above example, the student applied the operation (or KC) "add variable coefficients" in line (1) of the equation.

In the next section, we will describe the main Algebra Intelligent Tutoring Systems and the tools and types of feedback they offer to help students solve algebra word problems in these two phases.

3. Algebra Intelligent Tutoring Systems

The field of ITS has shown significantly improvements since the emergence of the first systems in the eighties (Woolf, 2009). The evolution of the Internet, the increasing performance of computers, and improvements of artificial intelligence techniques and tools have furthered the development of ITSs in several domains, such as Physics, Math, Medicine and others (see (Woolf, 2009) for an overview).

Previous work has largely been applied in classroom settings, demonstrating they can improve student performance on standardized and experimenter-designed tests by one-half to two standards deviation (Cohen et al., 2008; Koedinger & Sueker, 1996; Shelby et al., 2000). Some of the most known research focused on the Algebra content domain. This is the case of Cognitive Algebra Tutor (previously PAT) (Koedinger & Sueker, 1996), Aplusix (Nicaud et al., 2006, Nicaud, Bouhineau, & Huguet, 2002), ActiveMath (Goguadze & Melis, 2008; Melis, Goguadze, Libbrecht, & Ullrich, 2009) and AnimalWatch (Birch & Beal, 2008; Cohen et al., 2008). We believe there are two main reasons for this. First, Algebra is a content domain (or task domain (Vanlehn, 2006)) in which a great number of students experience poor achievement (Carpenter, Kepner, Corbitt, Lindquist, & Reys, 1982; National Commission on Excellence in Education, 1983). Secondly, it requires less effort to formalize math content into computer algorithms, because it is mainly composed of procedural content, which can be easily represented by computer algorithms. In the end of this section, we describe the main algebraic tutors proposed by the Artificial Intelligence and Education community.

AnimalWatch (Birch & Beal, 2008; Cohen et al., 2008) assists middle school students in solving arithmetic word problems. The mathematical problems explore environmental issues around endangered animal species. The main goal is to motivate students by connecting math to real problems involving animals. The main screen of AnimalWatch is composed of a textual description of a word problem, additional illustrations of the problem (such as video, charts, diagrams and pictures) and an answer box. The student solves the problem elsewhere (for instance, on paper) and provides the final result in the answer box (one step in the tutor interface). AnimalWatch is mainly concerned with the Problem Representation. Thus, it does not have an ES that is able to correct the problem in an intermediary step of the solving process. If student needs help, the tutor provides hints, but it is not able to identify which steps of the student's problem solving are incorrect if the student's answer does not match the expected result.

ActiveMath (Goguadze & Melis, 2008; Melis et al., 2009) is a web-based intelligent learning system for math that, in addition to algebra, teaches differential calculus, logics, statistics. It allows the students to freely navigate around the content. It uses a semantic web representation in order to choose learning objects for students, which can be textual explanations, several types of exercises (multiple-choice, word problem), and others. Like AnimalWatch, ActiveMath is more concerned with the outer loop, i.e. using a student model and web semantic information to decide which task is presented to the student. It does not incorporate an ES that supports students in the inner loop.

Aplusix and Cognitive Tutor do include an ES that is responsible for providing step-by-step guidance to the student during a *task.* During the inner loop, a tutor can offer customized mentoring in one or both of the problem solving processes explained in Section 2 (i.e. Symbolization or Problem Solution). Cognitive Tutor gives the student a linear or quadratic word problem to solve with the help of charts and a table that assist students in the Symbolization. It also has a simple editor for students to solve the linear equation.

On the other hand, the main interface of Aplusix is an editor where students can solve math problems in the fields of numerical calculations, expansions, factorizations, quadratic equations, inequalities, and systems of equations. It is not an ITS in the sense that it does not retain information about the skills that a student mastered and it does not choose the next task based on a student model (outer loop). For each step provided by the student, Aplusix is able to provide real time feedback by showing whether or not the answer is correct or by demonstrating how to solve the equation.

Both expert systems (Aplusix and Cognitive Tutor (CT)) were implemented as rule-based expert systems. This is also the case of the proposed expert system. The main differences among these works are the information associated with the rules and the interface options, which enable (or not), more powerful modes of interaction and feedback for the student. The ES of the Cognitive Tutor is also called Cognitive Model, since it is based on a cognitive approach to learning. It contains rules that represent the procedural knowledge of the student (Anderson, Corbett, Koedinger, & Pelletier, 1995; Ritter, Anderson, Koedinger, & Corbett, 2007). Generally, each rule symbolizes an operation (a knowledge component) necessary to solve an equation. However, the interface of the solver limit what students can do by making them choose one operation from a menu. Afterwards, students simply have to complete the new textboxes presented by the system. It does not allow the students to apply their own paper-and-pencil reasoning steps and strategies during the problem solving. One limitation of Aplusix is that it does not assess student knowledge. Table 1 summarizes the inner loop's main problem-solving functionalities supported by Aplusix and Cognitive Tutor. It is important to observe that the focus of this paper is on problem solving. For this reason, we are not taking into account functionalities provided in the Symbolization process.

Table 1
Inner loop functionalities supported by Algebra ITSs.

Supported functionalities	Cognitive Tutor	Aplusix	Pat2Math
Minimal feedback on steps	x	x	x
Error-specific feedback		x	x
Hints on the next step		x	x
End-of-problem review of the solution		x	x
Assessment of knowledge	x		x

The three ITSs provide minimal feedback on steps, i.e., they provide real-time correction in the form of a symbol (label, color or sign) that indicates whether or not a student response is correct. If the student did not correctly solve a step, the system should provide some specific feedback indicating how to solve the step. This is done by the Hint module of Pat2Math and by the companion agent in Aplusix. Although Cognitive Tutor presents this option for the Symbolization process, it does not work in the solver tool. Another function available in CT and Pat2Math is the assessment of student knowledge. In the CT, a Bayesian network assesses a student's skills and displays them in the "skillometer". Although Pat2-Math is able to track a student's performance in each knowledge component (currently using the arithmetic mean), it exhibits the student's general performance in a colored bar.

In the next section, we describe, in more detail, the main artificial intelligence techniques used in the implementation of ES in Algebra tutors.

4. Implementing the expert system of an algebra tutor

Artificial Intelligence classifies knowledge into two types: declarative and procedural (Nilsson, 1982). Procedural knowledge is concerned with understanding how to do things, such as, when to apply the distributive operation in a quadratic equation. Declarative knowledge is concerned with facts – for example, knowing that the capital of Brazil is Brasília. Generally, the Domain Knowledge module of an ITS is able to represent both of them.

These two types of knowledge usually require different forms of representation (Davis, Shrobe, & Szolovits, 1993). A usual representation format for procedural knowledge is a production rule. A rule is typically expressed as IF-THEN statements that denote: (1) the antecedent (IF) that embodies a set of conditions to be satisfied; (2) the consequent (THEN) that contains actions to be executed or new knowledge to be produced if the consequent is true. An example of a rule is:

```
IF the temperature is below 0
THEN the weather is cold
```

Rules are usually used by expert systems to represent procedural knowledge and allow powerful reasoning and inference. Besides having a knowledge base (KB) that stores permanent rules and facts, a Rule-Based Expert Systems (RBESs) also has a Working Memory that stores the facts and rules that are being applied in the current problem.

However, the most important component of a RBES is the inference engine. It selects those rules whose antecedents match Work Memory facts and stores them in the Conflict Set (also called the Agenda). If the Conflict Set has more than one rule (i.e., if more than one rule can be triggered), the inference engine uses a conflict resolution strategy to determine which rule to trigger. A RBES is a powerful reasoning mechanism when there is not a pre-specified sequence of actions to solve the problem, such as in an exhaustive search or in the case of Algebra, where students can solve a problem in several ways.

The inference engine is the most important component of a RBES and the most difficult one to implement. On the other hand, it is independent of the domain. Although the facts and rules are exclusive for each domain, the inference engine depends only on the language used for knowledge representation. For this reason, there are some implementations of the inference engine that assist the development of a RBES. They are called Expert Systems Shells. Drools is an ES shell implemented in Java (Bali, 2009; JBOSS, n.d.).

The PATEquation expert system is a production system implemented in Drools, version 4.0 (JBOSS, n.d.). We chose the shell Drools because it allows easy integration with Java. It is possible to invoke Java methods from Drools and vice versa. This enables us to develop more powerful and flexible rules.

In the next sections we will present more details about Pat2Math.

5. PAT2Math

In this section, Pwe present the ITS PAT2Math and the functionalities it offers to assist students in learning algebra.

PAT2Math stands for Personal Affective Tutor to Math. The main goal of this project is to develop a web-based ITS that takes into account students emotions (inferred by student's actions in the system interface and by facial expression (Jaques, Vicari, Pesty, & Martin, 2011)) when scaffolding, and information about skills they mastered, as an ITS usually does. The domain of Algebra, more specifically the content of linear and quadratic equations, was chosen because it usually induces fear and other negatives emotions in students that generally impair the learning process and result in frustration and increased dropout rates (Frenzel, Pekrun, & Goetz, 2007; Hannula, 2002).

Although, there are well-known ITS in literature (cited on Section 3), we chose to develop another Algebra ITS because most of the existing systems (such as Aplusix and Cognitive Tutor) are not open-source. We cannot extend or improve on them to integrate the emotions inference. Our intention is to make PAT2Math available without charge to all schools in Brazil. Our second goal is to develop an ITS that follows the algebra curriculum specified by the Brazilian Ministry of Education (Brasil. Secretaria de Educação Fundamental, 1997). Since all Brazilian schools must follow this specification curriculum, this would increase the acceptance of our ITS in Brazil.

The project began on 2008 and we have since developed PATequation, a tool that includes an editor and a powerful expert system that assists students in solving first and second degree equations. This tool comprises the inner loop of PAT2Math for problem solving and offers the services detailed in Table 1.

Due to space limitations, the focus of this paper is on the expert system and the rational functioning of the tutor. Some results of the ongoing research on the emotional aspects of the interaction student-tutor, such as recognition of students' emotions and expression of emotions by the tutor can be found at (Jaques, Lehmann, & Pesty, 2009; Jaques et al., 2011; Motola, Jaques, & Axt, 2008).

In the next section, we describe PATequation. It is an editor the student can use to solve equations step-by-step with immediate feedback. It implements the inner loop of PAT2Math. Currently, PATequation assigns tasks (outer loop) in a fixed sequence previously defined by a professor.

6. PATequation

PATequation is a Java applet that assists students in solving linear and quadratic equations in PAT2Math. Fig. 1 illustrates the interface of PATequation.

PATequation is comprised of an Editor (Fig. 1) where students exercise problem solving and an Expert System that provides step-by-step guidance for students. In Fig. 1, on the green blackboard, the first line (labeled A) contains the equation the tutor selected for the student to solve. The next lines represent step-by-step solutions provided by the students. The tutor offers immediate feedback for each step, indicating whether or not a response is correct by using thumbs up and thumbs down symbols. For each step, the student should choose an operation to apply and also provide the resulting equation.

The student chooses an operation by clicking a button with the corresponding symbol from one of the right panels. The selected operation symbol will appear in the step line on the left side of the equation (see letter D in Fig. 1). This allows the tutor to evaluate two things: (1) whether the students knew which operation to use to solve the problem and (2) whether they applied the operation correctly. It is not possible to evaluate the former skill if the

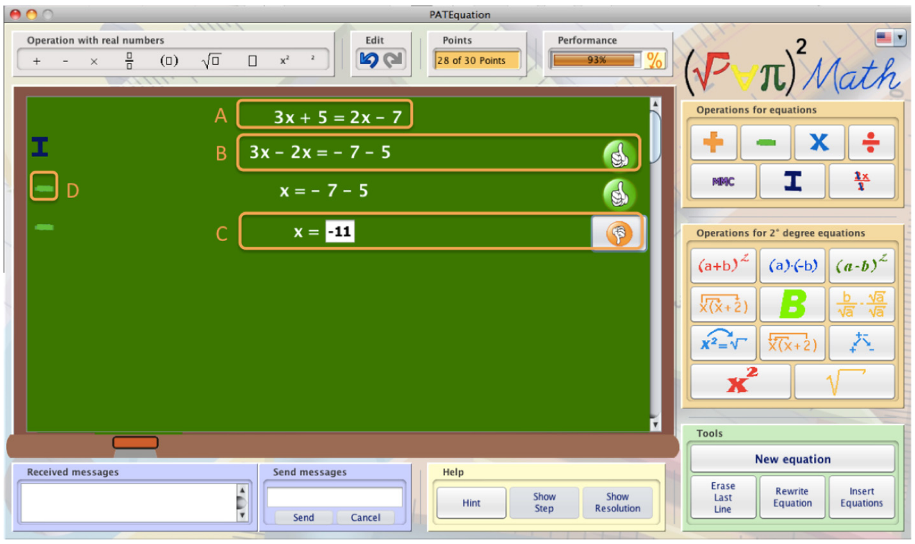

Fig. 1. PATequation interface. PATequation is a Java applet that assists students when solving linear and quadratic equations in PAT2Math. It provides the student with step-by-step guidance.

tutor interface does not ask students to explicitly choose an operation to apply. To allow advanced learners to operate freely, PATequation permits the student to provide a step solution that uses several operations. For example, for the equation $3x + 5 = 2x - 7$ (see Fig. 1), a student may provide as the next step $x = -7 - 5$ instead of creating three steps: $3x - 2x + 5 = -7$, $3x - 2x = -7 - 5$, $x = -7 - 5$. In this case, the system allows the student to choose anyone of the operations applied in the three steps. The tutor notifies students when they reach the final solution, i.e., when they find the value of the variable.

At any time during the problem-solving process, the students can also choose one of the following assistance options: (1) They can ask for a hint, (2) They can request that the tutor solve the next step of the equation, or (3) They can request that the tutor complete the equation from the current step.

PAT2Math has all of the functionalities described above (immediate feedback, hint, solve the next step and solve the full problem). In the next section, we describe in more detail the implementation of this expert system.

7. PATequation expert system

In order to provide immediate feedback for student steps during the problem solving process, PATequation must be able to solve linear and quadratic equations. Additionally it must be able to verify that the partial solution provided by the student matches one of the equations obtained by the tutor. Although PATequation has both of these features, the features represent different modules of an ITS and involve different types of rules. The former is called the *step generator* because it generates the next step for the student (Vanlehn, 2006). It also provides the next-step hint. The latter is called the *step analyzer* because it compares the step provided by the student with all possible steps found by the *step generator*. It then classifies the student's solution as either "correct" or "incorrect" (Vanlehn, 2006). It is this combination of the step generator and the step analyzer that we refer to as the "expert system". When it is necessary to make explicit which set of rules we are mentioning, we refer to them as Step Generator (SG) or Step Analyzer (SA), respectively. Fig. 2 represents the architecture of PAT2Math Expert System.

PATequation uses 53 rules for the SG and 21 rules for the SA. In the SG, each rule generally represents an algebraic operation to solve a linear or quadratic equation. They are knowledge components in the Algebra domain. For example, in order to solve linear equations, students can use one of the following operations: basic arithmetic operations (add, subtract, multiply or divide) on real numbers or on variables; a transformation (add, subtract, multiply or divide both sides of the equation); simplification of fraction, or least common multiple. In order to solve quadratic equations, they can apply the following operations in addition to the previous ones: factorization, expansion, quadratic formula, square of the sum of two numbers, difference of the square formula, square of the difference, rationalization, potentiation, and square root. Of the 53 rules available in the SG, 20 rules aim at replicating students' misconceptions (described bellow in more detail). The remaining rules are used to rewrite the equations (for instance, combine like terms) and other complementary operations (find the final roots for quadratic equations).

The expert system is invoked by PATequation's graphical interface when students request a demonstration of how to solve an equation (by clicking either the "Show Step" button or the "Show Resolution" button shown in Fig. 1) or when they want to verify if the step they provided is correct. When a student asks the tutor to show the equation solution, the ES calls the SG, which receives the equation and solves it from the current step. To verify if the student step is correct, the ES receives the equation along with the student's current step and sends them to the SA. The SA requests that the SG solve the equation and then verifies whether the SG is able to calculate the same step provided by the student. If it cannot, the SA requests that the SG solve the equation using the rules that represent students' misconception. If it does not recognize the step provided by the student, the SA gives the student a hint. Otherwise, the SA simply indicates that the step is incorrect.

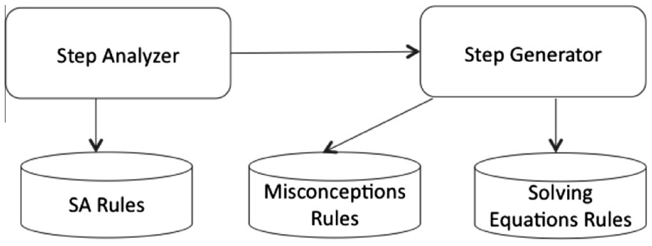

Fig. 2. The Architecture of PAT2Math Expert System. PAT2Math's expert system comprises two modules and three rules databases. The *step generator* generates the next step of the solution. The *step analyzer* compares the step provided by the student with all the possible solutions found by the *step generator*. It then classifies the student's solution as either "correct" or "incorrect".

7.1. Representation format for equations

In order for the ES to solve equations, it is necessary that the data structure to represent equations is easy to operate on and is unambiguous. For example, consider the equation $2x + 3 * x = 6$. It is ambiguous unless one knows that multiplication happens before subtraction. A natural data structure for representing an equation is a binary tree because it defines the order in which operations happen. For this reason, compilers have largely used it to represent mathematical expressions (Aho, 2007). A binary tree that represents an equation is called an "expression tree". Fig. 3 shows examples of expression trees for the equations $2x + 3x = 6$ and $(2x + 3) * x = 6$, respectively.

There are three types of nodes in an expression tree:

- *constant nodes*: hold real or integer numbers,
- *identifier nodes*: denote variables,
- *compound nodes*: represent the application of a operator to two operands, each one being another expression tree.

As Fig. 3 illustrates, the constant nodes and the identifier nodes are always leaf nodes (a node that has no child). On the other hand, the compound nodes have either one (unary operation) or two (binary operation) children trees, which represent the operands of the operation.

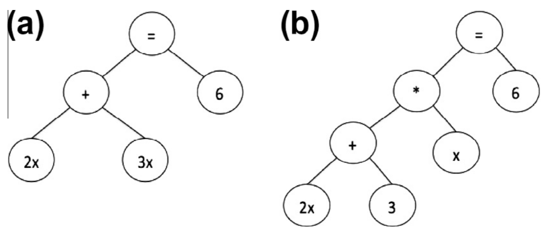

Fig. 3. Examples of an expression tree. (a) Is the expression tree for the equation $2x + 3x = 6$. (b) Represents the equation $(2x + 3) * x = 6$. The expression tree is the data structure that PAT2Math ES uses to represent equations.

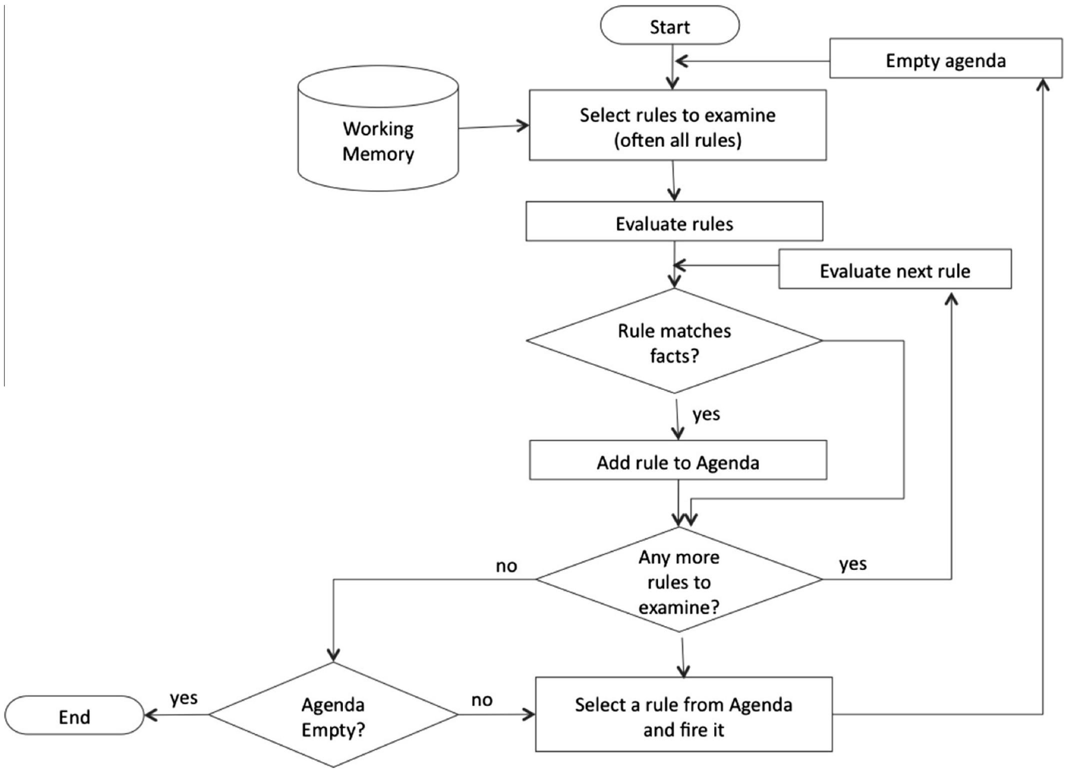

Fig. 4. An example of ES reasoning cycle based on (Hopgood, 2000). For every cycle, the ES selects from the knowledge base those rules that match with facts in the working memory and stores them in the Agenda. It triggers a rule from the Agenda according to a determined selection strategy. It also empties the Agenda and starts a new cycle until the Agenda has no more rules to trigger.

7.2. PATequation step generator

In a rule-based expert system, the inference engine (IE) executes several cycles of reasoning to solve problems. In each cycle, it executes the following actions (shown in Fig. 4). First, it selects, from the knowledge base, the rules that correspond to facts in the working memory and stores them in the Agenda, a special memory area that stores that rules selected to be triggered. Among these rules, it chooses one rule to trigger according to a determined strategy of selection. For example, in PATequation, each rule has a priority number and the IE selects the rule with highest priority to trigger. In the case where all the rules have the same priority, the IE triggers the rule that has matched those facts most recently added to the database. If the rule produces new knowledge, the IE adds this new fact to the working memory. It also empties the Agenda and starts a new cycle until the Agenda has no more rules left to trigger.

There are two reasoning mechanisms in the RBES inference engine: Forward Chaining (FC) and Backward Chaining (BC). Forward Chaining selects rules whose antecedents match facts in the working memory while Backward Chaining examines the rules consequents. While the FC is more appropriate for diagnostic problems, the BC is more suitable for proving assertions. BC is more efficient because it only triggers the rules that are necessary to solve the problem.

As previously explained, the PATequation ES was implemented with the Drools shell. The Drools IE applies Forward Chaining. This is the most appropriate type of reasoning for algebra problem solving because the solution is previously unknown.

Indeed, there are several paths to solve an equation. Fig. 5 illustrates the available paths to solve the equation $4x + 4 = 9 - x$. It is represented as a directed graph, in which the nodes denote a step solution. Each edge represents the operation that is applied to the equation in the predecessor node in order to obtain the equation in the successor node.

Even when solving a very simple equation, such as the aforementioned equation in Fig. 5, there are two possible paths.[1] When demonstrating how to solve an equation, PATequation should be able to choose a solution path. This is implemented by applying higher priority to the rules that represent operations with greater significance. An operation can have high significance because it has higher precedence or because its application is more desirable for pedagogical reasons. For example, in Fig. 5, although both paths are correct, the left path is the more desirable because teachers generally prefer to first isolate the variable on the left of the equation in order to emphasize it.

In the forward chaining reasoning, the first step of the IE is selecting the rules for examination. Generally, it selects all the rules of the KB. It evaluates the condition part of each rule. If the condition component is satisfied, it adds the rule to conflict set. After examining all the rules, it selects one conflict set rule to trigger according to a conflict resolution strategy. For example, the strategy could select the rule with higher priority. Finally, the IE updates the working memory with the new facts generated by the rule and empties the conflict set in order to start a new cycle.

The following scenario, illustrated in Table 2, shows how PATequation solves the euation $4x + 4 = 9 - x$ through forward chaining reasoning. First, it translates the equation from the infix math notation to postfix (Goodrich & Tamassia, 2010) in order to convert it into an expression tree (Lafore, 2003). It includes the expression tree as a new fact into the working memory of Drools and launches the inference engine. At this moment, the FC reasoning begins. Generally every cycle solves one step of the equation.

In the first cycle, the IE examines all rules in the KB in order to

[1] Actually, there are other paths if we allow recursive operations.

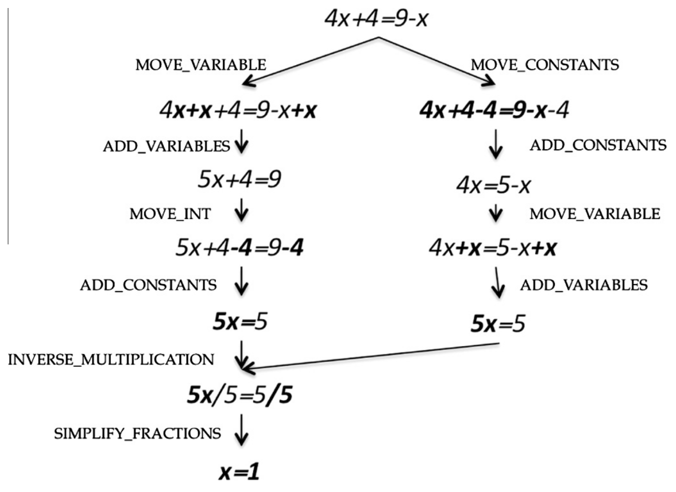

Fig. 5. Paths to solve the equation $4x + 4 = 9 - x$. There can be several paths to solving an equation. A path is a sequence of steps with each step resulting in an intermediate equation. A step is a user interface action (for example, to simplify the fraction $5x/5$ in an intermediate equation) that the student takes in order to achieve a task[2]. (In fact, Vanlehn differs step and learning events, where the step is a user interface action and the learning event is the application of a knowledge component to achieve a task; it is just a mental event. For the sake of simplicity, we just refer to steps in this article to mean that it is a user action that involves one or more learning events.) This figure illustrates two possible paths to solve the equation $4x + 4 = 9 - x$. The left and right paths have both six steps (the two last steps are equal).

select those rules whose antecedents contain conditions that are satisfied by the current equation. Because the operation of addition is only applied on the same side of the equation, this rule is not eligible. The IE chooses the rules MOVE_INT and MOVE_VARIABLE, which aim at moving the operand from one side of the equation to the other by applying an operation on both sides. While the goal of the MOVE_INT rule is to move constants to the right side, the purpose of MOVE_VARIABLE rule is to move all variables to the left side. Both rules aim at isolating the variables on the left and constants on the right side. The IE selects and triggers the MOVE_VARIABLE rule because it has a higher priority. The resulting equation ($4x + 4 + x = 9 - x + x$) replaces the previous one in the Working Memory. This prevents the IE from applying the operations recursively.

In the second cycle, there are two possible operations that the IE can apply: (1) sum the variables on the left and right sides of the equation and (2) apply MOVE_INT. MOVE_VARIABLE is not selected because it does not satisfy the condition that the equation only present one variable term on its right side. The IE selects the rules representing these two operations and inserts them into the Agenda. The former rule has higher priority and is triggered. The resulting equation ($5x + 4 = 9$) is now the only fact in the working memory. In fact, the IE applies this rule twice. In the second cycle, it adds $4x + x$ on the left side of the equation. In the third cycle, it adds $-x + x$ on the right side.

In the fourth cycle, there is only one rule eligible to be triggered, the MOVE_INT rule. The IE triggers this rule and a new equation ($5x + 4 - 4 = 9 - 4$) replaces the previous one in the Working Memory.

In the fifth and sixth cycles, it selects and triggers the rule ADD_CONSTANTS twice. Each cycle produces the equations $5x = 9 - 4$ and $5x = 5$, respectively.

In the next cycle (7th), there is only one rule whose conditions match with the fact $5x = x$ in the working memory. This rule divides both sides of the equations by 5 in order to cancel the coefficient of the term $5x$. The resulting fact is $5x/5 = 5/5$.

In the following two cycles, the IE applies the rules to simplify the fractions on both side of the equation. The result is that the equation $x = 1$ is the only fact left in the Working Memory.

In the last cycle, the IE triggers the rule whose purpose is to test if the final response was achieved, i.e., PATequation found the final value of the equation variable. In this case, it removes all the facts of the Working Memory. This stops the execution of the IE.

Besides the cycles above cited, there are other cycles in which the IE applies auxiliary rules aimed at preparing the equation for the next operation. For example, there is a rule based on the commutative property of numbers that aims at grouping the similar terms in the equation. It transforms the equation $x + 4 + 3x = 0$ into the equation $x + 3x + 4 = 0$. As the objective of this type of rule is simply to prepare the expression tree for an operation, we do not cite it in the above scenario because we want to focus on the rules that actually represent a real step towards solving the problem.

In this section, we demonstrated how PATEquation ES solves an equation. This functionality allows the system to show students how to solve a step or the full equation from any point of the solving process when they assistance. If the students ask only for the next step, only one cycle of the forward reasoning is performed. If the student asks for the full solution, the IE iterates through as many cycles as necessary to solve the equation. It also shows all the facts generated in each solving cycle. In both cases, the initial fact is the current equation.

In the next Section, we show how the Step Analyzer verifies students' steps in order to provide immediate feedback.

7.3. PATequation step analyser

As we previously mentioned, in order for a learning system to be considered an intelligent tutoring system it must be able to provide immediate feedback to the students in the inner loop (Van-

Table 2
Illustration of the working memory and agenda for solving equation $4x + 4 = 9 - x$.

#	Working memory (facts before firing rule)	Working memory (facts after firing rule)	Agenda (rules)	Triggered rules
1	$4x + 4 = 9 - x$	$4x + 4 + x = 9 - x + x$	(1) MOVE_INT (2) MOVE_VARIABLE	MOVE_VARIABLE
2	$4x + 4 + x = 9 - x + x$	$5x + 4 = 9 - x + x$	(1) ADD_VARIABLES (2) MOVE_INT	ADD_VARIABLES
3	$5x + 4 = 9 - x + x$	$5x + 4 = 9$	(1) ADD_VARIABLES (2) MOVE_INT	ADD_VARIABLES
4	$5x + 4 = 9$	$5x + 4 - 4 = 9 - 4$	(1) MOVE_INT	MOVE_INT
5	$5x + 4 - 4 = 9 - 4$	$5x = 9 - 4$	(1) ADD_CONSTANTS	ADD_CONSTANTS
6	$5x = 9 - 4$	$5x = 5$	(1) ADD_CONSTANTS	ADD_CONSTANTS
7	$5x = 5$	$5x/5 = 5/5$	(1) DIVIDE_BOTH_SIDES	DIVIDE_BOTH_SIDES
8	$5x/5 = 5/5$	$x = 5/5$	(1) SIMPLIFY_FRACTION	SIMPLIFY_FRACTION
9	$x = 5/5$	$x = 1$	(1) SIMPLIFY_FRACTION	SIMPLIFY_FRACTION
10	$x = 1$	⟨EMPTY⟩	(1) TEST_END	TEST_END

```
(1) The SA executes the SG in order to obtain the next step of the
equation;
(2) If there is no other possible solution, it returns the linked list;
(3) If there is a solution:
     (a)  It stores the current equation and the operation applied in a
     linked list;
     (b)  It compares the student equation (stdEq) and the resulting
     equation:
          (i)   If they are equivalent, it returns the linked list.
          (ii)  Otherwise, it recursively calls the current function
          passing the resulting equation as the curEq parameter.
```

Fig. 6. PATequation step analyser algorithm. The SA verifies students' steps in order to provide immediate feedback.

lehn, 2006). Additionally it should be able to verify which operations they used correctly and which they used incorrectly. The Step Analyzer (SA) accomplishes this task, which is also called Model Tracing (Anderson, Corbett, Koedinger, & Pelletier, 1995; Heffernan et al., 2008).

The SA is a recursive function that receives as parameter the current equation (*curEq*) and the student step (*stdEq*) and returns a linked list containing the sequences of operations necessary to achieve the student's answer. The pseudo-code below demonstrates how this recursive function works.

The recursive algorithm in Fig. 6 allows backtracking, i.e., the IE can return to a previous state in order to try another solution path. While backtracking is a powerful process, it can significantly decrease the system performance. Without using any heuristic to minimize the possible paths, the FC performs a full breadth-first search, resulting in a worst case runtime complexity of $O(n^d)$, where n is the maximum branching factor of the search tree (in an ES the number of possible rules) and d is the depth of the last-cost solution (number of triggered rules) (Russell & Norvig, 2003). In PATequation, for the worst cases, the value of n can be 53, which are the numbers of rules available in the SG set.

In order to minimize the number of unnecessary triggered rules, we create meta-rules that infer the operation that the student applied. These additional rules compare the current equation and the student step in order to identify which operations the student most likely applied. For example, let us assume that the current equation is $1 + 2 + 3 = x$ and the student step response is $1 + 5 = x$. The meta-rules verify that the terms 1 and x are still present in the student equation, but not the terms 2 and 3. They also verify that there is an *add* operator between these terms. In that case, it tries to apply the rule ADD_CONSTANTS for these two operands in order to try to find 5. With the meta-rules, the time complexity is reduced to O (d), where d is the number of triggered rules, it means, the number of rules that need to be triggered in order to achieve the solution. It happens because the meta-rules guide the FC to trigger only the rules that are necessary to find the student's solution.

In fact, these meta-rules try to emulate a backward chaining in the expert system, which is the most appropriate reasoning for this type of problem. Backward reasoning only triggers the rules that are necessary to solve the problem. However, Drools and other stable expert systems shells do not implement this type of reasoning. This is the reason why we decided to implement this meta-rules approach.

Another potential problem with expert systems is the number of comparisons that need to be made between rules and facts in the database. Most current ES shells use a RETE algorithm for this match process, which reduces the average complexity from O (n^d) to O (n^2), where n is the number of facts in the Working Memory (Watson, 2008). As we generally work with few facts (generally two) in PATequation, this process does not impair its performance.

8. Evaluation

In order to evaluate the effectiveness of our tool, we compared the students' improvement in performance in a group of 22 students who solved equations in PATequation (experimental group-EG) in relation to a group of 21 students who only exercised without the tool (control group-CG).

The hypothesis of the experiment is that PATequation helps the student in learning linear equations providing a statistically higher gain scores (difference between posttests and pretests) compared to students who exercised equation without the tool. The null and alternative hypotheses are:

H_0: EGgain $\leqslant$ CGgain (null hypothesis)
H_1: EGgain > CGgain (alternative hypothesis)

The null hypothesis is that the gain scores in the EG is less than or equal to the CG gain. Moreover, the claim is that the EG gain was statistically higher than the CG gain, identifying that students exercising with PATequation learned more than students who only solved equations in paper-and-pencil.

The experimental evaluation took place in June 2012 in a private school in Porto Alegre, Brazil. Forty-three students from two seventh grade classes of an elementary school participated in the experiment. Because the students were minors (12–13 years), we asked the parents to sign the informed consent form before conducting our study. This evaluation can be classified as a quasi-experiment, because the sample subjects were not chosen randomly (Campbell & Stanley, 1963). These classes were selected because of their previous contact with our research group and also because of the interest of their teachers.

Fig. 7 illustrates the phases of the evaluation experiment. Initially, the subjects were informed about the objectives of the experiment (evaluate PATequation), as well as the voluntary and confidential nature of their participation. The experiment was conducted in 4 sessions each lasting for 50 min, totaling 3 h and 20 min of interaction. In the first lesson, we administered a pretest aimed at identifying the students' prior knowledge. During the following two classes, the students in the experimental group solved equations with our tool and students in the control group used only paper and pencil. In the last class they solved the posttest and completed the questionnaire. The questionnaire consisted of four questions regarding students' habits when studying math and thirteen questions about the students' experience with PATequation after having used it (Fig. 8 shows the students' answers for one of the questions).

The students' grades and the mean and standard deviation of pretest and posttest in control and experimental groups are shown in Table 3. Each test was composed of 5 problems for students to solve, worth a total of 2.5 points. The equations in each test were different, but evaluated the same skills. Table 4 shows the mean and standard deviation for pretests and posttests scores.

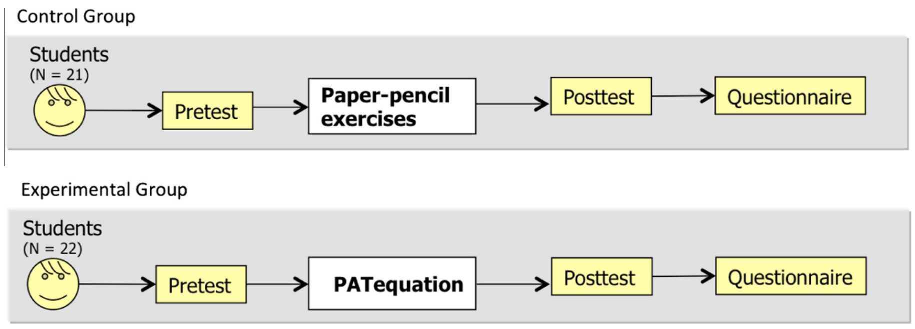

Fig. 7. The PAT2Math evaluation experiment. The experiment consisted of 3 phases: (1) pretest, (2) solving equations in paper-and-pencil for the control group, or solving equations in PAT2Math for the experimental group, and (3) posttest and questionnaire.

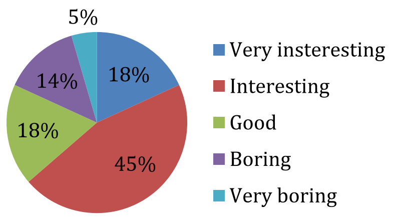

Fig. 8. Students' answers to the question "Solving equations in PATequation was . . ." After completing the posttest, students were asked to complete a questionnaire regarding their math habits and their experience with PATequation.

Table 3
Students' pretest and posttest scores.

Control Group				Experimental Group			
Student	Pretest	Posttest	Gain	Student	Pretest	Posttest	Gain
1	1.7	2.0	0.3	1	1.0	2.0	0.3
2	0.4	1.0	0.6	2	1.0	1.9	0.6
3	1.0	1.8	0.8	3	2.0	2.5	0.8
4	2.0	2.0	0.0	4	1.5	2.2	0.0
5	2.3	2.5	0.2	5	2.5	2.5	0.2
6	2.5	2.5	0.0	6	2.5	2.5	0.0
7	1.3	2.5	1.2	7	2.0	2.5	1.2
8	1.4	2.5	1.1	8	1.5	1.7	1.1
9	0.9	2.0	1.1	9	2.0	2.2	1.1
10	2.0	2.2	0.2	10	2.0	2.0	0.2
11	1.2	1.7	0.5	11	1.0	2.5	0.5
12	1.5	1.7	0.2	12	2.0	2.5	0.2
13	1.0	2.0	1.0	13	1.5	2.5	1.0
14	2.0	2.0	0.0	14	1.5	2.1	0.0
15	1.1	1.5	0.4	15	0.5	2.5	0.4
17	2.1	2.1	0.0	17	1.0	2.1	0.0
18	1.7	1.9	0.2	18	0.5	2.1	0.2
19	0.3	0.5	0.2	19	2.0	2.5	0.2
20	0.5	0.5	0.0	20	2.0	2.5	0.0
21	2.0	2.4	0.4	21	1.0	1.6	0.4
				22	2.0	1.9	0.3

We used a one-tail *t*-test because we did not know the population standard deviation (we only know the sample *s*). With a confidence level of 90% ($\alpha = 0.1$), we obtain $p = 0.06$ ($t = -1.5376$, df = 39). The mean score increased from 1.445 ($sd = 0.642$) on the pretest to 1.865 ($sd = 0.599$) on the posttest. The difference between the two means is statistically significant at the .1 level. Thus, as $p < \alpha$, we reject the null hypothesis and conclude that there is evidence to say with 90% confidence that students using PATequation have higher gain scores in equation solving than students using pen and pencil.

On the questionnaire, 22% of students reported it was very interesting to solve equations in PATEquation, 56% of them stated it was interesting and for 22% of students, it was good. Finally, 52% of students agreed that solving equations in PATequation with the help of hints increased their knowledge about algebraic equations.

Table 4
Students' pretest and posttest mean and standard deviation (sd).

	Pretest	Posttest	Gain
Control group			
Mean	1.44	1.86	0.42
sd	0.64	0.59	0.40
Experimental group			
Mean	1.6	2.2	0.7
sd	0.59	0.29	0.56

9. Conclusions

In this paper we presented PAT2Math, an expert system module of an Algebra ITS. The ES is responsible for correcting student steps and modeling student knowledge components during problem solving in the PAT2Math's editor (PATequation). Another important function of this module is to demonstrate to the student how to solve a problem. In order to provide this step-by-step guidance in the inner loop, ES is able to solve the same problems that student do.

The proposed ES was implemented as rule-based expert system in which one rule is used to represent each algebra operation a student should master. The ES is composed by meta-rules that aim at reducing the number of possible paths to explore when correcting a student's solution. This solution reduces the computational complexity from $O(n^d)$ to $O(d)$, where n is the number of rules in the knowledge base and d is the number of rules triggered.

In this paper, we mainly focused on the expert system of PAT2-Math, describing its implementation and its approach on solving equations and providing feedback to the user. We also described a quasi-experiment with forty-three 7th grade students, which demonstrated that the performance improvement observed in students using PATequation to solve equations was statistically significant. Encouraged by our initial results we plan to conduct a new user study involving more students over a longer period of time. We expect to gain new insights about the long-term applicability and scalability of our approach.

Acknowledgements

This research is supported by the following research funding agencies of Brazil: CAPES, CNPq and FAPERGS.

We greatly appreciated the feedback on the article from Sven Stork and Meghan Tighe.

References

Aho, A. V. (2007). *Compilers: Principles, techniques, and tools*. Pearson/Addison Wesley, p. 1009.

Aleven, V., McLaren, B. M., Sewall, J., & Koedinger, K. R. (2009). Example-tracing tutors?: A new paradigm for intelligent tutoring systems. *International Journal of Artificial Intelligence in Education, 19*(2), 105–154.

Anderson, J. R., Corbett, A., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences, 4*(2), 167–207.

Bali, M. (2009). *Drools JBoss Rules 5.0: Developer ' s Guide* (p. 320). PACKT.

Birch, M., & Beal, C. R. (2008). Problem posing in AnimalWatch: An interactive system for student-authored content. In *International Florida Artificial Intelligence Research Society Conference* (pp. 397–402).

Brasil. Secretaria de Educação Fundamental (1997). *Parâmetros Curriculares Nacionais: matemática* (Vol. 3, p. 148). Brasília: MEC/SEF.

Campbell, D. T., & Stanley, J. C. (1963). In N. L. Gage (Ed.). *Experimental and quasi-experimental designs for research* (Vol. 20, pp. 84). Rand McNally. http://dx.doi.org/10.1016/0306-4573(84)90053-0.

Carpenter, T. P., Kepner, H. S., Corbitt, M. K., Lindquist, M. M., & Reys, R. E. (1982). Student performance in algebra: Results from the national assessment. *School Science and Mathematics, 82*(6), 514–531. http://dx.doi.org/10.1111/j.1949-8594.1982.tb10052.x.

Chaachoua, H., Nicaud, J. F., Bronner, A., & Bouhineau, D. (2004). APLUSIX, A learning environment for algebra, actual use and benefits. In *Proceedings of ICME-10* (p. 8). Copenhagen, Denmark.

Cohen, P. R., Beal, C. R., & Adams, N. M. (2008). The design, deployment and evaluation of the AnimalWatch intelligent tutoring system. *Information Systems, 178*, 663–667.

Davis, R., Shrobe, H., & Szolovits, P. (1993). What is knowledge representation? *AI Magazine, 14*(1), 17–33.

Frenzel, A., Pekrun, R., & Goetz, T. (2007). Girls and mathematics—A "hopeless" issue? A control-value approach to gender differences in emotions towards mathematics. *European Journal of Psychology of Education, 22*(4), 497–514.

Gama, C. A. (2004). *Integrating metacognition instruction in interactive learning environments*. University of Sussex.

Goguadze, G., & Melis, E. (2008). Feedback in ActiveMath exercises. In *Proceedings of international conference on mathematics education* (pp. 1–7).

Goodrich, M. T., & Tamassia, R. (2010). *Data structures and algorithms in Java*. Wiley.

Hannula, M. (2002). Attitude towards mathematics: Emotions, expectations and values. *Educational Studies in Mathematics, 49*(1), 25–46.

Heffernan, N. T., Koedinger, K. R., & Razzaq, L. (2008). Expanding the model-tracing architecture?: A 3 rd generation intelligent tutor for algebra symbolization. *Artificial Intelligence*, 18.

Hopgood, A. A. (2000). *Intelligent systems for engineers and scientists. Library*. Boca Raton: CRC Press (p. 461).

Jaques, P. A., Lehmann, M., & Pesty, S. (2009). Evaluating the affective tactics of an emotional pedagogical agent. *ACM symposium on applied computing* (Vol. 1, pp. 104–109). Hawaii: ACM. http://dx.doi.org/10.1145/1529282.1529304.

Jaques, P. A., Vicari, R., Pesty, S., & Martin, J.-C. (2011). Evaluating a cognitive-based affective student model. In S. K. D'Mello, A. C. Graesser, B. Schuller, & J.-C. Martin (Eds.). *International conference on affective computing and intelligent interaction (ACII)* (Vol. 6974, pp. 599–608). Springer.

JBOSS (n.d.). Drools Expert. <http://www.jboss.org/drools/drools-expert.html> Retrieved 03.06.12.

Koedinger, K. R., Anderson, J. R., Hadley, W. H., & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education, 8*(1), 30–43.

Koedinger, K. R., & Sueker, E. L. F. (1996). PAT goes to college: evaluating a cognitive tutor for developmental mathematics. In *Proceedings of the 1996 international conference on Learning sciences* (pp. 180–187). International Society of the Learning Sciences.

Lafore, R. (2003). *Data structures and algorithms in Java*. Sams.

Mayer, R. E. (1999). *The promise of educational psychology: Learning in the content areas*. Merrill.

Melis, E., Goguadze, G., Libbrecht, P., & Ullrich, C. (2009). ActiveMath–A learning platform with semantic web features. *The Future of Learning*, 1–22 (2006).

Motola, R., Jaques, P. A., & Axt, M. (2008). A Domain and Platform Independent Architecture for Presentation of Affective Behaviors of Animated Pedagogical Agents. In *Anais do Simpósio Brasileiro de Informática na Educação* (pp. 22–31). Porto Alegre: SBC.

Munem, M. A., & West, C. (2003). *Beginning algebra*. Kendall/Hunt Publishing Company.

National Commission on Excellence in Education (1983). *A nation at risk: The imperative for educational reform*. Washington, DC.

Nicaud, J. F., Bittar, M., Chaachoua, H., Inamdar, P., & Maffei, L. (2006). Experiments with Aplusix in four countries. *International Journal for Technology in Mathematics Education, 13*(2), 79–88.

Nicaud, J. F., Bouhineau, D., & Huguet, T. (2002). The Aplusix-Editor?: A new kind of software for the learning of algebra. In *International Conference on Intelligent Tutoring Systems* (pp. 178–187). Biarritz, France: Springer-Verlag.

Nilsson, N. J. (1982). *Principles of artificial intelligence*. Springer.

Polya, G. (2004). *How to solve it: A new aspect of mathematical method*. Princeton University Press.

Ritter, S., Anderson, J. R., Koedinger, K. R., & Corbett, A. (2007). Cognitive tutor: applied research in mathematics education. *Psychonomic Bulletin and Review, 14*(2), 249–255.

Russell, S. J., & Norvig, P. (2003). *Artificial intelligence: A modern approach* (2nd ed.). Pearson Education.

Shelby, R., Schulze, K., Treacy, D., Wintersgill, M., Vanlehn, K., & Anders Weinstein, A. (2000). An assessment of the andes tutor. In *Physics education research conference*. Rochester, NY.

Vanlehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education, 16*(3), 227–265.

Watson, M. (2008). *Practical artificial intelligence programming with Java*.

Woolf B. P. (2009). *Building intelligent interactive tutors. Pragmatics* (p. 467). Elsevier.